Learning to Detect Features in Texture Images

Linguang Zhang Szymon Rusinkiewicz
Princeton University

Abstract

Local feature detection is a fundamental task in computer vision, and hand-crafted feature detectors such as SIFT have shown success in applications including image-based localization and registration. Recent work has used features detected in texture images for precise global localization, but is limited by the performance of existing feature detectors on textures, as opposed to natural images. We propose an effective and scalable method for learning feature detectors for textures, which combines an existing "ranking" loss with an efficient fully-convolutional architecture as well as a new training-loss term that maximizes the "peakedness" of the response map. We demonstrate that our detector is more repeatable than existing methods, leading to improvements in a real-world texture-based localization application.

1. Introduction

Many computer vision tasks require computing local features as the first step. These tasks include but are not limited to image alignment [33, 3], image retrieval [17, 9], image-based localization and reconstruction [24, 25]. A pipeline used for computing local features given a single input image typically consists of a **feature detector** and a **feature descriptor**, which are often performed sequentially. For both of these two important components, there is a substantial amount of work on designing hand-crafted solutions, and some of the best-performing hand-crafted pipelines such as SIFT [11] have become gold standards in real applications. Nevertheless, nearly all existing hand-crafted solutions are optimized for and evaluated on natural images. There are, however, many more types of images that are outside the scope of natural images, but also require a well-performing feature pipeline. A recent project named Micro-GPS [31] has demonstrated that precise global localization can exploit *textures* (such as those present on ground surfaces ranging from carpet to asphalt), because textures globally exhibit many distinctive and persistent features that can be used for unique identification. While a hand-crafted feature pipeline such as SIFT indeed works properly in most of the textures demonstrated in that work, its robustness varies on different textures. We observe that different textures consist of different "basic elements" that are not always ideal for a particular feature detector. For example, a typical blob or corner detector will find few features on the stripes present in a wood texture. In this paper, we propose a learning-based feature detector optimized for texture images, and demonstrate its effectiveness in the Micro-GPS system(Figure 1).

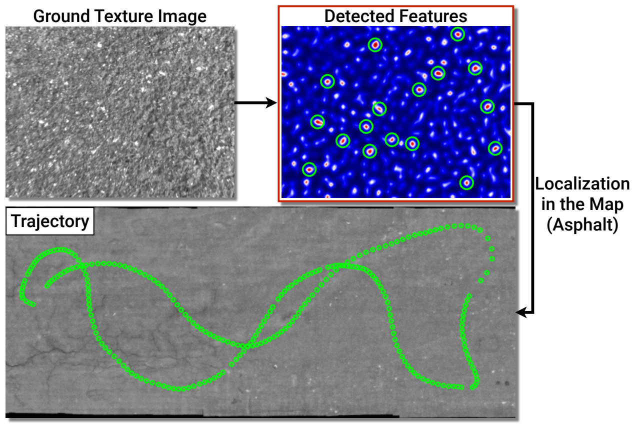

Figure 1. From each test image, our proposed detector extracts highly repeatable features, which can be utilized by Micro-GPS to achieve precise global localization in a pre-built map, such as the asphalt map being shown. Note that Micro-GPS locates each test image *independently* in the map (ignoring temporal coherence).

In recent years, machine learning techniques have shown success in improving feature pipelines. Most of this previous work, however, focuses on learning feature descriptors [23, 5, 22, 30], which we argue is a more straightforward problem than learning a feature detector. While it is difficult to obtain ground-truth labels for both feature detection and feature description, learning a feature descriptor often relies on *self-supervision*, which is usually achieved by training with corresponding and non-corresponding image patches. Exploiting such self-supervision is more natural for learning a feature descriptor, because it exactly matches how a descriptor is evaluated — minimizing the distance between corresponding descriptors and maximizing the distance between non-corresponding descriptors. In contrast, it is less obvious how to perform similar self-supervision when learning a feature *detector*, because the criteria used for evaluating a detector are very different. In general, a detector is considered well-performing if the interest points output by the detector 1) can be repeatedly detected even when the image undergoes certain transformations such as rotation; 2) are locally distinctive, which means that they cannot be easily confused with other nearby points; and 3) are sufficiently numerous to be useful in applications such as retrieval, registration, and localization. The combination of these criteria cannot be easily translated into a loss function to be used in (self-supervised) training.

Recent work has demonstrated that a feature detector can indeed be learned in an unsupervised manner [10, 20] without using existing feature detectors for supervision. Specifically, these methods attempt to output a "response map" with each pixel indicating how interesting the pixel is, and the only constraint used in these methods is that the response map should be *consistent* under certain criteria when the image undergoes transformations. Because this leads to only weak constraints, some work has proposed to add supervision by, for example, constraining the response map to fire on points output by a hand-crafted detector [32]. Although this approach was shown to work, it is limited by the effectiveness of the underlying TILDE detector [26], which our experiments show has only moderate performance for the texture images on which we focus.

We propose an approach to feature detection that retains self-supervision, augmenting the goal of consistency (as expressed by the "ranking loss" of Savinov et al. [20]) with the desire to make the response map as "peaked" as possible. This leads to more localized local extrema in the response map, which in turn boosts the repeatability of the detector. Our detector also utilizes a fully-convolutional network architecture with a large receptive field, leading to both high efficiency and suitability for a wide variety of textures. Finally, and most crucially, we demonstrate that textures are sufficiently varied that optimal performance for each type of texture can be achieved by training on that texture alone. While we do evaluate the repeatability of detectors trained on one texture and tested on another, our main focus is on answering: *given sufficient amount of images of a specific texture, is it possible to learn a "perfect" feature detector for this texture alone?*

The major contributions of this paper are:

- Proposing a fully-convolutional network architecture that can be efficiently applied on a full-sized image without separate evaluation on each pixel.
- Describing a method to maximize the *peakedness* of the response map and proving that it is critical to improving the repeatability of the learned detector.
- Evaluating design choices, demonstrating that maximal effectiveness requires a large receptive field but is relatively insensitive to the tuning of other parameters.
- Applying the learned detector in a real localization application that requires features with high repeatability.
- Making our implementation (training and testing) available for future work.

2. Related Work

While classic features detectors were hand-crafted using a human-specified definition of "interestingness", recent ones use learning to improve performance. These either select a subset of points output by a hand-crafted detector or learn a definition of "interestingness" from scratch.

2.1. Hand-Crafted Feature Detectors

There is extensive work on designing a feature detector that performs well on natural images. Detecting corners is one of the earlier strategies [6, 13]. Alternatively, one can detect "blobs". The SIFT detector [11] uses Differences of Gaussians (DoG) to approximate the Laplacian-of-Gaussian (LoG) filter, and looks for local extrema over scale and space. SIFT has shown great robustness in real-world applications and remains a gold standard. A major limitation of SIFT is its speed. While one can use GPUs to accelerate SIFT [27], SURF [2] approximates LoG using a box filter and significantly speeds up detection. As an alternative to SIFT, MSER [12] detects blobs by extracting covariant regions from the image and fitting ellipses to these regions. In addition to detecting blobs, SFOP [4] also detects junctions. WADE [18] demonstrates that salient symmetries can be leveraged to detect repeatable interest points even in images related to untextured objects, which are difficult cases for a corner or blob detector.

2.2. Learned Feature Detectors

FAST [16] achieves fast corner detection, and machine learning techniques are applied to accelerate detection. By minimizing the pose estimation error for stereo visual odometry, one can learn a convolutional filter (LCF) for feature detection [15]. A classifier can be learned from SIFT features surviving matching tasks, and combining the classifier with the original SIFT detector helps achieve better matchability [7]. TILDE [26] uses stack of pre-aligned images undergoing drastic brightness changes and learns a detector to predict highly repeatable SIFT keypoints across images. Similarly, LIFT [28] also uses patches corresponding to SIFT keypoints to train their feature detector. Lenc and Vedaldi [10] show that it is possible to train a feature detector using the covariant constraint only, by forcing the network to output covariant transformations given an image patch and its transformed version. This work was extended by using features detected by TILDE as guidance [32].

Savinov et al. [20] propose to learn a detector by ranking image patches. This method is based on the assumption that if a patch has a higher score than another patch in the response map, this ranking relationship should remain unchanged when the image is transformed. While this method shows improvement over previous detectors, and we incorporate its ranking loss into our work, it has two major limitations. First, the network is underconstrained because only ranking loss is used. Guaranteeing a low ranking loss does not necessarily imply high repeatability of the detector, and we show that this is data-dependent. Second, all network architectures demonstrated in the original paper are constrained by a small receptive field (17×17 kernel in their work). The small receptive field is inadequate for many of the textures we consider. In addition, the deep convolutional architecture, which shows the best performance in the cross-modal detection task, requires per-pixel traversal in the test image. We ameliorate these problems by incorporating a peakedness loss term that effectively improves repeatability, and using an efficient fully-convolutional network that can be directly applied to the test image.

2.3. Global Localization using Ground Textures

The recent Micro-GPS work [31] has shown that images of ground textures can be used for global localization. The key observation that drives the system is that seemingly homogeneous or random textures exhibit landmarks that can enable unique identification. Even in some highly repeating textures where landmarks may look very similar, spatial arrangements of the landmarks can also serve as useful visual cues. The Micro-GPS system consists of a downward-facing camera and a image processing unit that quickly locates the input image in the database. To build the database, texture images are densely captured in a target area and stitched into a globally consistent map. Within each captured image, SIFT features are extracted, compressed and indexed into a compact database that can be stored on a mobile device. During testing, SIFT features are extracted from the test image and matched against the database. The matched features later go through an efficient Hough voting procedure, and inliers are used for pose estimation.

Once the database is built, the original images could be safely deleted. Therefore the size of the database only depends on the number and dimensionality of the features stored. The number of features can also affect the matching performance because as the feature space becomes denser, matching becomes not only slower but also more inaccurate due to the larger number of false positives. In the original paper, the authors specify that keeping 50 SIFT features per image is sufficient to guarantee reasonably good performance on across all types of textures. Therefore further improvement of Micro-GPS is mainly constrained by feature detection. Due to the setup of Micro-GPS, there are only three requirements that the feature detector needs to satisfy: high repeatability, rotation invariance, and global distinctiveness. Scale or lighting invariance is not required here, because the images used by Micro-GPS are captured at a constant height (constant scale) and with generally stable illumination. In this work, we show that by plugging in a highly repeatable feature detector learned using our approach, Micro-GPS can achieve significantly better robustness compared with the original SIFT-based version.

We focus on a range of ground textures displaying a variety of phenomena. In general, they contain some "basic element" that can be leveraged by a feature detector, but a hand-crafted detector designed to handle a single type of feature is not expected to work consistently well on all textures. Examples of the texture images are shown in first row of Figure 5. Below we briefly discuss the visual characteristics of the ground textures.

Carpet: The carpet texture is normally the least challenging texture for a feature detector, since it contains knots with a regular arrangement.

Asphalt: The asphalt texture is mainly random, with stains and cracks that can be used as distinctive features.

Wood: The wood texture is usually the most challenging because most detectors are not tuned to recognize its major visual cue: the striped grain pattern.

Tiles: The tile texture contains randomly embedded color chips that are easy to recognize, but a detector needs to have large a receptive field to capture the chips.

Granite: The granite texture does not contain obvious distinctive elements, but local brightness variations can be used for unique identification.

Concrete: Similarly to the asphalt texture, the concrete texture contains black stains. However, the size of the stains varies significantly.

Coarse: The coarse asphalt texture contains large, distinctive stones, making it relatively easy for a detector.

3. Approach

Defining the desired appearance of a feature in texture images is a non-trivial task because even humans cannot reliably label repeatable features. Hand-crafted detectors have one definition of "feature," and fail to work when textures do not fit that definition. For example, wood textures often contain stripes that cannot be identified by a blob detector or a corner detector. Since there is no obvious way to obtain labels to enable supervised learning, we choose to perform unsupervised training, following the basic idea of Savinov et al. [20].

3.1. Feature Detection by Ranking

The network we train will take as input an image, and produce as output a *response map*. Each value in the response map indicates how likely this pixel is to be a distinctive interest point. Specifically, we would like the deep-neural-network to learn a scoring function $\mathcal{F}(.)$ that maps the input image patch to a single-valued score. We want

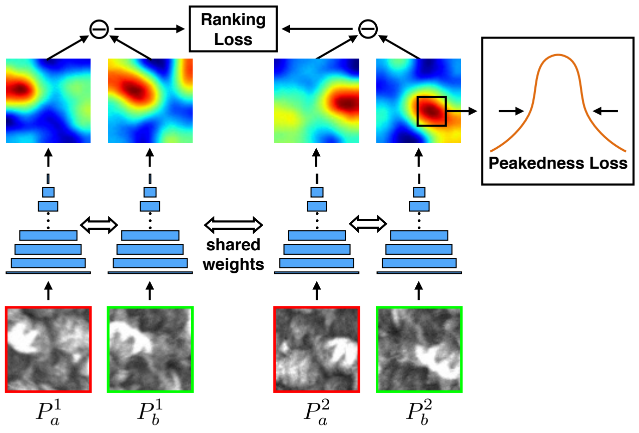

Figure 2. Illustration of the training pipeline. The network is pre-trained with the ranking loss only, and tuned using both the ranking loss and the peakedness loss.

this score to be consistent under transformations: if one point is more "interesting" than another, then it should still be more interesting when the image is transformed. That is, given any pair of patches $\{P_a^1, P_b^1\}$ and their randomly transformed versions $\{P_a^2, P_b^2\}$, we want the scoring function to satisfy either of the following two cases:

$$\begin{cases} \mathcal{F}(P_a^1) > \mathcal{F}(P_b^1) \text{ and } \mathcal{F}(P_a^2) > \mathcal{F}(P_b^2) \\ \mathcal{F}(P_a^1) < \mathcal{F}(P_b^1) \text{ and } \mathcal{F}(P_a^2) < \mathcal{F}(P_b^2) \end{cases} . \quad (1)$$

Combining into a single inequality:

$$\mathcal{R} = \left(\mathcal{F}(P_a^1) - \mathcal{F}(P_b^1)\right)\left(\mathcal{F}(P_a^2) - \mathcal{F}(P_b^2)\right) > 0. \quad (2)$$

This inequality ensures consistent ranking of P_a and P_b, and we apply a hinge loss to obtain our *ranking loss* term:

$$\mathcal{L}_{\text{rank}}(P_a^1, P_a^2, P_b^1, P_b^2) = \max(0, M_{\text{rank}} - \mathcal{R}), \quad (3)$$

where M_{rank} is the margin and correlates to the confidence of ranking consistency. We set $M_{\text{rank}} = 1.0$ throughout our experiments. In each training iteration, the network takes in two pairs of corresponding image patches which are randomly rotated. An illustration of the ranking network is shown in Figure 2.

In selecting an appropriate architecture for the ranking network, we take into consideration both performance and efficiency. Performance is strongly affected by the size of the receptive field. While one could let the network see a larger area by downsampling the original image, significant downsampling erases subtle details. We show that while a small window can indeed work in some "easy" texture images, a larger window is critical to better adaptability. Our network has a receptive field of 65×65, leading to improved performance relative to the original networks of Savinov et al. [20], which have a receptive field of size 17×17. When considering efficiency, we notice that the best-performing architecture (Deep convolutional network) used by Savinov et al. requires per-pixel traversal of the input image. We instead use a fully-convolutional network containing only convolutional layers and rectified linear units (ReLU), with no pooling or padding. This architecture is efficient because the trained model can be directly applied to the whole image, yielding a complete response map.

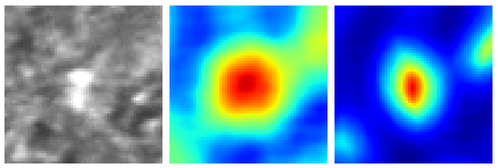

Figure 3. From left to right: zoomed-in view of a stain in the asphalt texture, response map output by the network trained using ranking loss only, and response map after optimizing the peakedness. The latter response map is more robust against noise when performing non-maximum suppression.

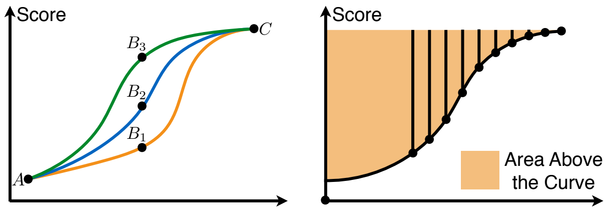

Figure 4. Illustration of response curves. Left: different response curves can lead to same ranking. Right: peakedness of the response curve can be evaluated as the area above the curve; specifically, for the k highest values in a local window, we sum up the difference between each value and the maximum.

3.2. Optimizing Peakedness of the Response

Detection purely by ranking has several limitations due to its unconstrained nature. First, the the loss is unchanged if the ranking of image patches is flipped. In theory, this is not a problem because we can consider both local maxima and local minima as good candidates for features. In practice, however, we observe that each training run results in a network in which either maxima or minima yield better (and more visually plausible) results. Therefore, we use a validation set to evaluate the repeatability of both maxima and minima, and we negate the response map if the minima perform better. In other words, we always end up with a response map whose local maxima are the features we use.

Another, more serious, limitation of a ranking network is that there are many functions that all result in the same *relative* rankings of different pixels, while yielding different repeatability in matching. For example, consider the patch in Figure 3. The response map shown at center is broad, meaning that the feature was not localized precisely. In contrast, the response map at right is more "peaky," which ultimately results in more accurate matching across images.

In order to encourage the network to learn the response map at right, we include a peakedness term in our training loss. This is based on looking at all the responses in a patch, and encouraging only a few of them to be large. For example, if we look at the responses within the neighborhood of a maximum, sorted from lowest to highest, different networks might produce any of the three curves in Figure 4, left. We prefer the lowest curve, since it will have the most-peaked response. This is accomplished by maximizing the *area above the curve*, shown in Figure 4, right: this forces the values in the neighborhood of the maximum to be as small as possible.

Specifically, during training we consider not just 65×65 input patches (which would result in a single output value per patch), but rather patches $\hat{P}$ of size (64+w)×(64+w). Because our scoring network $\mathcal{F}$ is fully convolutional, this results in an output response map of size $w \times w$, where w =7 in our experiments. We sort the scores in this map to obtain a response curve $\tilde{\mathcal{F}}(\hat{P})$. The peakedness of the patch is computed as the difference between the maximum and the average of the largest k scores:

$$\gamma(\hat{P}) = \max\big(\tilde{\mathcal{F}}(\hat{P})\big) - \frac{1}{k}\sum_{i}^{k} \tilde{\mathcal{F}}(\hat{P})_i. \tag{4}$$

The *peakedness loss* function forces the peakedness to be above a certain margin:

$$\mathcal{L}_{\text{peak}}(\hat{P}) = \max\big(0, M_{\text{peak}} - \gamma(\hat{P})\big), \tag{5}$$

where M_{peak} is the desired peakedness (3.0 in our experiments). We compute the total training loss by averaging the peakedness loss of the four input patches, and combining with the ranking loss:

$$\mathcal{L}_{\text{total}}(\hat{P}_a^1, \hat{P}_a^2, \hat{P}_b^1, \hat{P}_b^2) = \mathcal{L}_{\text{rank}}(P_a^1, P_a^2, P_b^1, P_b^2) + \alpha \cdot \tfrac{1}{4}\big(\mathcal{L}_{\text{peak}}(\hat{P}_a^1) + \mathcal{L}_{\text{peak}}(\hat{P}_b^1) + \mathcal{L}_{\text{peak}}(\hat{P}_a^2) + \mathcal{L}_{\text{peak}}(\hat{P}_b^2)\big), \tag{6}$$

where $\{P_a^1, P_a^2, P_b^1, P_b^2\}$ are the center 65×65 patches of $\{\hat{P}_a^1, \hat{P}_a^2, \hat{P}_b^1, \hat{P}_b^2\}$ and α is a weighting parameter used to balance the ranking loss and the peakedness loss. Note that within each batch, not all the patches strongly correlate to good features, thus it can be harmful to maximize the peakedness for patches with relatively weak responses. Therefore we exclude the 25% of patches with the smallest maximal scores when computing the peakedness loss.

3.3. Implementation

Datasets: The datasets used in Micro-GPS consist of texture images densely captured along overlapping paths and registered to each other. Previous works usually train their networks using patches cropped around SIFT features that survive a structure-from-motion pipeline [28], because this is perhaps the most efficient way to generate corresponding image patches. Savinov et al. [20] use the DTU Robot Image Dataset [1], for which ground truth 3D points are available, but such a dataset is difficult to acquire and its size is small considering the broad space of natural images. The advantage of the texture datasets we use is that generating a pair of corresponding patches is as simple as cropping around any point in the overlapping region, and, more importantly, these patches are not biased to any existing feature detector. We use one region in each texture for training and the others for validation and testing. We only crop image patches from image pairs with an overlapping area over 40%. For training, we randomly crop 512k pairs of corresponding image patches. Each patch is cropped with random orientation, which aims to make the scoring network rotation-invariant. In each training iteration, the quadruple used is constructed by randomly combining two pairs of corresponding patches.

Training: Our network consists of 12 convolutional layers, with the first 11 followed by a ReLU activation: details are provided in supplemental material. We train our network on an NVIDIA M40 GPU using the PyTorch framework [14], using the Adadelta algorithm [29] to minimize loss. The batch size is 256 and the network is trained using only the ranking loss for 10000 iterations. The network is then tuned with both ranking and peakedness loss for 2000 iterations. We observe that this schedule leads to better performance than training from scratch with both losses. We also observe that batch normalization blurs the response map and lowers repeatability, so we omit it. Training the network typically takes three hours.

Feature Detection in a Test Image: Given a test image of arbitrary size H×W, we first reflection-pad the image by 32 pixels on each side, because this is how much our network removes from each border. Applying the network to the padded image results in an output response map of size H×W. We then follow the general interest point localization pipeline detailed in [11]. Gaussian blurring with σ =2 is performed before non-maximum suppression, which effectively prevents multiple detections in a small neighborhood. To limit the number of output interest points, we simply select the n largest local maxima based on score. Finally, we apply sub-pixel localization based on the second-order Taylor expansion of the scoring function to refine the integer-valued interest point locations.

Computational Efficiency: The proposed architecture is efficient, as compared to state-of-the-art methods that require evaluating each pixel separately (i.e., running the network on the neighborhood around every pixel, independently). Our full pipeline runs at 2.5fps (1288×964). In comparison, QuadNet [20] (our implementation) runs at 0.00775fps with batch processing, while our network computed independently per pixel (as opposed to over the entire image at once) would run at 0.00408fps due to a larger re-

Table 1. Number of repeatable features detected by each method (across 20 images, keeping a maximum of 200 features per image).

Method	**carpet**	**asphalt**	**wood**	**tile**	**granite**	**concrete**	**coarse**
HarrAff	394	350	34	706	223	495	466
HarrLap	390	351	35	702	224	494	468
HessAff	447	247	507	84	233	149	799
HessLap	444	249	490	126	230	150	803
MSER	590	475	68	1556	404	952	799
FAST	1708	1524	1017	2276	1574	1802	1583
SIFT	2022	1474	610	1510	1385	1869	1456
SURF	2451	2879	1455	2271	2496	2738	2125
LCF	811	779	740	1068	817	795	1093
SFOP	2504	1831	1074	2328	1993	2219	1989
WADE	2898	2296	20	1230	2254	1870	1992
TILDE-P24	2029	2185	2380	2770	2607	3244	2093
TILDE-P	1777	1869	2182	2625	2329	3039	1975
Linear17	3550	3106	1567	2730	3143	3304	3182
DCNN17	3589	3304	2260	2650	3079	3006	2755
Pretrained	3290	3332	1892	3383	3384	2716	3188
Tuned	**3715**	**3344**	**2906**	**3397**	**3431**	**3607**	**3314**

ceptive field. Hand-crafted detectors such as SIFT are faster: the detector portion of SiftGPU [27] runs at 25fps.

4. Results

4.1. Evaluation Protocol

The test set we used for evaluating repeatability is generated by randomly sampling pairs of overlapping images with an overlap area above 50%. However, the images in the texture dataset tend to have similar orientation, which is not sufficient for evaluating rotation-invariant feature detectors. We therefore randomly rotate one image in each pair and recompute the transformation between the two images. Given two images and the transformation matrix between them, we run the feature detector on each image, select features in the overlapping region, and count detected features as "repeatable" if they are found in both images, within 5 pixels after applying the transformation.

For a fair evaluation, we must ensure that methods are not rewarded for finding too many features (in which case finding a matching feature by chance would be too easy) or too few (in which case the matching percentage may be high even though the number of detected features may be too low for many applications). We therefore keep only the 200 strongest features (if that many exist) in the overlapping region for each image, similarly to what has been done in Zhang et al. [32]. We perform bidirectional matching to prevent multiple features from matching to the same feature in the other image. Finally, we report the *number* of matches, rather than the *fraction* of matching features, to penalize methods that could not detect at least 200 features (in original images of resolution 1288×964 and 1280×720).

4.2. Performance

We compare our detector to hand-crafted detectors including SIFT [11], SURF [2], MSER [12], WADE [18], SFOP [4], Harris Laplace [13] (HarrLap), Hessian Laplace [13] (HessLap), Harris Affine [13] (HarrAff) and Hessian Affine [13] (HessAff). We also compare our detector to learned detectors including FAST [16], LCF [15], TILDE [26] (TILDE-P, TILDE-P24), and Quad-networks [20] (Linear17, DCNN17). The TILDE detector is trained using time-lapse image sequences, thus we directly use their trained model. TILDE-P24 is an approximation of TILDE-P. Linear17 stands for the one-convolutional-layer network that regresses a 17×17 patch to a single-valued score. DCNN17 stands for the deep-convolutional network which was originally designed for cross-modal detection; it requires per-pixel traversal of the input image.

We also attempted to retrain the covariant detector [10] and its extended version assisted by "standard patches" using the publicly available code [32], but training did not converge, even on the relatively simple carpet texture. This might be due to the significant difference between natural images used by their methods and texture images. We observe that the image patches cropped from natural images have obvious variations in appearance, but the patches cropped from texture images appear similar to each other.

Table 1 shows the total number of repeatable features detected using the above detectors (more detailed statistics can be found in the supplemental material). Each detector is applied to 20 pairs of images for each type of texture, which means that at most 4000 repeatable features can be retained. As expected, none of the hand-crafted detectors perform consistently well on all types of textures, which motivates the need for a texture-specific detector. Similarly to the original Quad-network detectors, our pretrained detectors also suffer from the drawbacks of using only ranking loss. However, on textures which contain large elements, such as the colored chips in the tile texture, our pretrained detector outperforms Quad-network detectors since our architecture has a much larger receptive field. After tuning our detectors by introducing the peakedness loss, our detectors outperform all other detectors on every texture, with particular gains on difficult textures such as wood. It is also worth pointing out that the TILDE detectors, which are trained using natural images, achieve reasonably good performance on all textures although surprisingly they perform worse on the easiest carpet texture.

The good performance of our tuned detectors can be better understood through the response maps shown in Figure 5. Although training using the ranking loss is sufficient on "easy" textures, the network tends to output smooth response maps for challenging textures such as wood and concrete. Adding the peakedness loss forces the network to output a sharper response map, making feature localization more robust to noise. Also note that the response map is reversed on the concrete texture, which explains the significant repeatability gain.

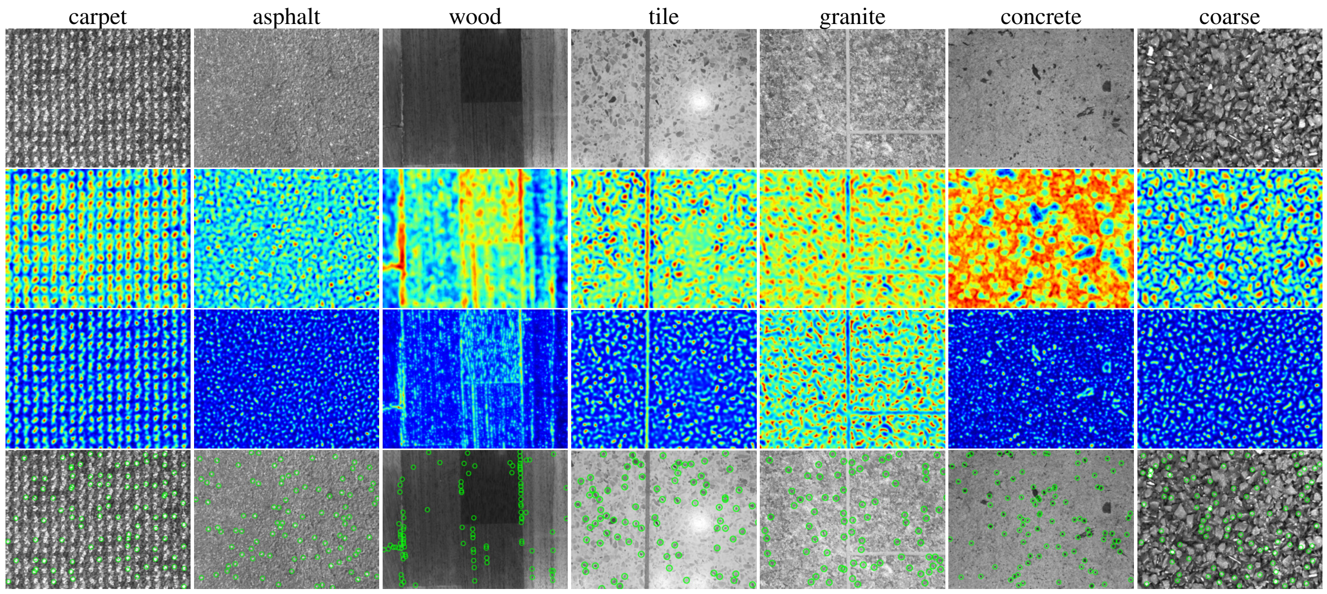

Figure 5. Top to bottom: input, response maps (ranking loss only), response maps (ranking and peakedness loss), top 200 features.

4.3. Impact of Parameters

There are three major parameters that can influence the performance of the detector: the weight assigned to the peakedness loss (α), the size of the window used to compute the peakedness loss (w), and the number of pixels used to compute the area above the curve (k). We investigate the effects of these parameters by, without loss of generality, beginning with α=0.5, w=7, and k=20, then varying each parameter independently around this configuration. For each parameter setting, we report how much repeatability the detector gains after the tuning stage.

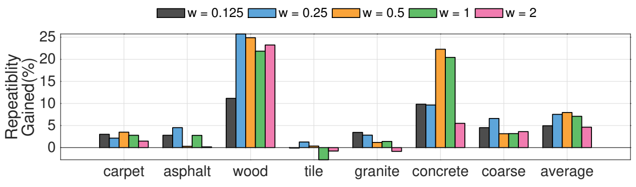

Figure 6. Repeatability gain of detectors tuned with different α.

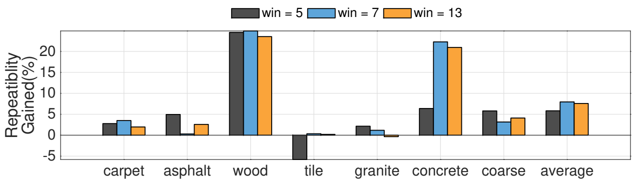

Figure 7. Repeatability gain of detectors tuned with different w.

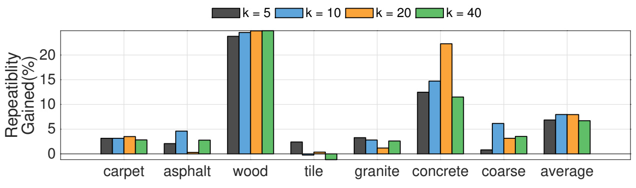

Figure 8. Repeatability gain of detectors tuned with different k.

Varying α: The weight assigned to the peakedness loss influences how much the network is willing to increase the ranking loss in order to improve the peakedness of the response function. Figure 6 shows that the tuned detector typically reaches the best performance when the weight is set to 0.5. Beyond this, repeatability often decreases because the detector starts to overlook the ranking loss.

Varying w: The network must see a sufficiently-large local window in order to learn to maximize local peakedness. Increasing window size too far, however, makes it more likely that the window will contain multiple peaks. We observe that using w=7 results in the best repeatability gain on average (Figure 7).

Varying k: It is often not a good idea to use all the pixels in a local window for computing the area above the curve. This is because the sorted scores only change drastically in the first k pixels. In Figure 8, we observe that using 10 or 20 pixels results in the best repeatability gain on average. Note that when using 20 pixels, the detector trained on the concrete texture yields superior repeatability.

4.4. Cross Evaluation

An important question one may ask is: *how does a detector trained on one texture perform on a completely different texture?* Another interesting question is: *can we train a good "universal" detector using the union of all texture images?* Below we show the cross evaluation result of both the pretrained models (Figure 9) and the tuned models (Figure 10). The universal model trained using the ranking loss only is close to unusable. This is likely because the network cannot find a consistent ranking across various textures. However, with the peakedness loss which only requires the

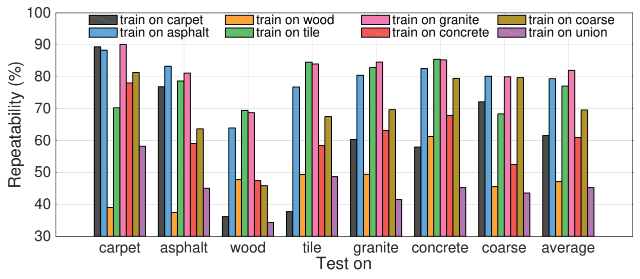

Figure 9. Cross evaluation result of the pretrained detectors.

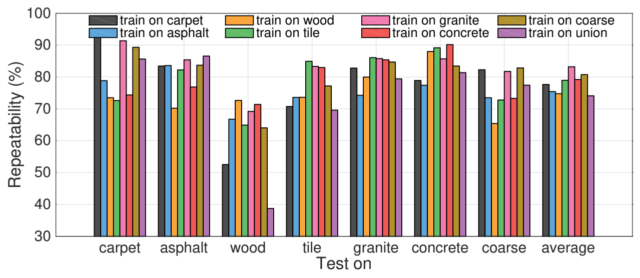

Figure 10. Cross evaluation result of the tuned detectors.

universal detector to optimize local maxima, the detector is able to achieve reasonably good repeatability, although the wood texture is still too challenging. Another interesting finding is that the detector trained on the granite texture performs much better than the universal detector on all types of textures. We observe that both local maxima and local minima found in the response map of the granite texture correspond to very good interest points, which implies that the detector trained on the granite texture potentially has better adaptability. This experiment partially explains why handcrafted features are still preferred to learned features in real applications in the domain of natural images [19, 21]: using more data for training does not necessarily help learning-based methods generalize well, and using a smaller set of representative data might be a better solution.

4.5. Effectiveness in a Localization Application

To evaluate the performance of the proposed detector in real applications, we combine our detector with the SIFT descriptor [11] and plug into the Micro-GPS [31] pipeline, which uses compressed 16-dimensional SIFT descriptors. The feature orientation required by SIFT descriptor is also computed using the orientation estimator of the SIFT pipeline. We compare the SIFT detector and our detector by sampling 50 and 20 features according to the response from each database image, respectively, to build the feature database. The success rate of the system demonstrates the usefulness (distinctiveness and repeatability) of the selected features. Figure 11 shows performance under different configurations. The original Micro-GPS clusters features based on their scales, and feature matching is performed within each scale group, which effectively improves the system performance (last bar in each column of Fig-

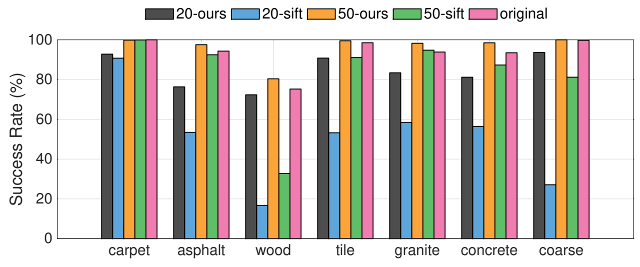

Figure 11. Performance on Micro-GPS.

ure 11). For a fair comparison, the feature scale detected by SIFT is not used, and the descriptor is computed for both SIFT and our detector using a fixed scale of 6.0. Our learned detector is a clear winner even with feature scale absent.

5. Conclusion and Future Work

We present a pipeline for training a feature detector specialized in detecting locally distinctive features in texture images. Explicitly defining what a good feature should look like in texture images is challenging, and so we choose to learn the scoring function in an unsupervised manner. We demonstrate that learning the scoring network purely through a simple ranking loss makes the response curve highly underconstrained, because the response curve corresponding to a desired ranking is not unique. We propose to tune the network by maximizing local peakedness, which significantly improves the repeatability on challenging textures. Moreover, the network architecture we use allows the network to see a much larger area than previous work while guaranteeing testing efficiency by avoiding pixel traversal. Lastly, we show that our detector outperforms SIFT in the "Micro-GPS" texture-based global localization application.

In the future, several immediate directions can be pursued. First, we would like to extend our approach to scale space using a spatial transformer [8]. We experimented with an approach from previous work based on image pyramids [20] and found that some apparently-good features are suppressed by failing to become local extrema in scale space. Second, a locally-distinctive interest point does not imply that the descriptor computed using the point is also sufficiently globally-distinctive for applications that involve feature matching. We believe that a feature descriptor can be introduced to guide feature detection.

Acknowledgements

We would like to sincerely thank Adam Finkelstein, Olga Russakovsky and Princeton Graphics Group for their helpful discussions. This work is supported by NSF grants IIS-1421435 and CHS-1617236.

References

[1] H. Aanæs, A. L. Dahl, and K. S. Pedersen. Interesting interest points. *International Journal of Computer Vision*, 97(1):18–35, 2012.

[2] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. *Computer vision–ECCV 2006*, pages 404–417, 2006.

[3] M. Brown and D. G. Lowe. Automatic panoramic image stitching using invariant features. *International journal of computer vision*, 74(1):59–73, 2007.

[4] W. Förstner, T. Dickscheid, and F. Schindler. Detecting interpretable and accurate scale-invariant keypoints. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2256–2263. IEEE, 2009.

[5] X. Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg. Matchnet: Unifying feature and metric learning for patch-based matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3279–3286, 2015.

[6] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Manchester, UK, 1988.

[7] W. Hartmann, M. Havlena, and K. Schindler. Predicting matchability. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9–16, 2014.

[8] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, pages 2017–2025, 2015.

[9] H. Jegou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. *Computer Vision–ECCV 2008*, pages 304–317, 2008.

[10] K. Lenc and A. Vedaldi. Learning covariant feature detectors. In *Computer Vision–ECCV 2016 Workshops*, pages 100–117. Springer, 2016.

[11] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[12] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and vision computing*, 22(10):761–767, 2004.

[13] K. Mikolajczyk and C. Schmid. Scale & affine invariant interest point detectors. *International journal of computer vision*, 60(1):63–86, 2004.

[14] A. Paszke, S. Chintala, R. Collobert, K. Kavukcuoglu, C. Farabet, S. Bengio, I. Melvin, J. Weston, and J. Mariethoz. Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration, may 2017.

[15] A. Richardson and E. Olson. Learning convolutional filters for interest point detection. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 631–637. IEEE, 2013.

[16] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. *Computer Vision–ECCV 2006*, pages 430–443, 2006.

[17] Y. Rui, T. S. Huang, and S.-F. Chang. Image retrieval: Current techniques, promising directions, and open issues. *Journal of visual communication and image representation*, 10(1):39–62, 1999.

[18] S. Salti, A. Lanza, and L. Di Stefano. Keypoints from symmetries by wave propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2898–2905, 2013.

[19] T. Sattler. Out with the old? convolutional neural networks for feature matching and visual localization.

[20] N. Savinov, A. Seki, L. Ladicky, T. Sattler, and M. Pollefeys. Quad-networks: unsupervised learning to rank for interest point detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

[21] J. L. Schönberger, H. Hardmeier, T. Sattler, and M. Pollefeys. Comparative evaluation of hand-crafted and learned local features. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[22] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer. Discriminative learning of deep convolutional feature point descriptors. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 118–126, 2015.

[23] K. Simonyan, A. Vedaldi, and A. Zisserman. Learning local feature descriptors using convex optimisation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(8):1573–1585, 2014.

[24] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3d. In *ACM transactions on graphics (TOG)*, volume 25, pages 835–846. ACM, 2006.

[25] N. Snavely, S. M. Seitz, and R. Szeliski. Modeling the world from internet photo collections. *International Journal of Computer Vision*, 80(2):189–210, 2008.

[26] Y. Verdie, K. Yi, P. Fua, and V. Lepetit. Tilde: a temporally invariant learned detector. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5279–5288, 2015.

[27] C. Wu. Siftgpu: A gpu implementation of scale invariant feature transform (sift)(2007). *URL http://cs. unc. edu/˜ ccwu/siftgpu*, 2011.

[28] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua. Lift: Learned invariant feature transform. In *European Conference on Computer Vision*, pages 467–483. Springer, 2016.

[29] M. D. Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[30] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1802–1811, 2017.

[31] L. Zhang, A. Finkelstein, and S. Rusinkiewicz. High-precision localization using ground texture. *arXiv preprint arXiv:1710.10687*, 2017.

[32] X. Zhang, F. X. Yu, S. Karaman, and S.-F. Chang. Learning discriminative and transformation covariant local feature detectors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6818–6826, 2017.

[33] B. Zitova and J. Flusser. Image registration methods: a survey. *Image and vision computing*, 21(11):977–1000, 2003.